ARP Failure

Debugging saga

Fri, May 28, 2004

This note describes a bug in the IRM system code that was found after a week-long investigation. The original symptom of failure was that the local application AAUX, which supports the Acnet ACNAUX protocol, failed to download the Acnet tables after a reboot.

Analysis of the symptom

That the tables were not downloaded was not a great concern and could mostly be overlooked, since the tables are stored in nonvolatile RAM and are seldom modified. In addition, they are downloaded every 12 hours just to keep them from becoming too stale. When using the Memory Dump page to examine the SM static memory structure allocated by AAUX for the duration of its activity, it was easy to see that the tables were not downloaded. In that case, modifying the 12-hour countdown value so that it reached zero sooner resulted in the tables being equally promptly downloaded. So, it knew how to download the tables and would do so every 12 hours, but it seemed to have trouble achieving this downloading soon after a reboot.

Another diagnostic that was useful for seeing what was happening after a reboot is the Net Frames page application, normally attached to Page F. It reads from the NETFRAME data stream, into which a record is written every time a network frame (datagram) is received or transmitted, listing up to 237 of the most recent records.

One more diagnostic that is useful is listing out, via the Print Memory page application, the contents of the ETHPQ, or ethernet output pointer queue, which is located at 0x19E000. Every message to be written to the network is recorded here. Each 8-byte entry includes a byte containing a "used" bit, a delay counter that times out 1 second, which is a requirement lest the entry be aborted, and a special bit that is set by the ethernet transmit interrupt routine. It also includes the message block type#, the target node#, and the address of the block containing the message contents. One can correlate the entries in this queue with the listing produced by Page F.

The failure to download the Acnet tables did not always occur; sometimes, the tables were downloaded successfully. Some seemingly random set of events resulted in success or failure.

Enter ARP

After a few days of examining these diagnostics, it became clear that an occasional failure to receive an ARP reply might explain the symptom. This could be seen by viewing the ETHPQ entry for the ARP request message sent to OPER, whose Acnet node# is 0x09EA. In fact, the ARP message entry was the first one recorded in the diagnostic records. When a reboot resulted in failure to download, there was no apparent reply to the ARP request; however, the ARP request must have been transmitted, because of the special bit that was set in the ETHPQ entry. Furthermore, it was possible to find the transmitted ARP request message block, and its contents looked correct.

As more background on the ARP logic found in IRMs, when a message is about to be queued to a target node for which no physical address is known in the IPARP table entry, the message block pointer is sidelined into a linked list that is connected to the incomplete IPARP entry for that node. The first time an entry is added to the empty linked list, an ARP request is queued to the network and is promptly transmitted. For any additional message that is to be queued to the same target node, its message block pointer is added to the linked list, but no additional ARP request is sent, since one is already pending. When an ARP reply is received for a node that has such a non-empty linked list, the IPARP entry is updated with the physical address copied from the ARP reply, and all messages in the linked list are written into the ETHPQ. The reason for all this logic is to allow network message processing with other nodes to proceed, while only delaying the transmission of

messages to nodes for which no physical address has yet been received. The lack of a network physical address for node "A" will not inhibit sending messages to any node "B" for which a physical address is known. (Before this logic was implemented in the IRM system code, sending a message to a node for which no physical address was known would result in sending an ARP request instead and the original message discarded. This behavior was very user-unfriendly.)

Examining what actually is to happen after a reboot, an Acnet protocol "ping" message is sent to OPER by AERS, the Acnet alarm reporter local application used in each IRM system, which does this during its own initialization logic. On the cycle following its initialization, it sends a front end boot (FEBT) message to OPER, where the Acnet alarm handler called AEOLUS resides. Finally after a delay of "n" 15 Hz cycles, where the value of n depends upon the local node#, AAUX sends a request for the first Acnet table, also targeting OPER. For the case of test node0509, the value of n is 27, or nearly 2 seconds. (In general, the value of n is the low byte of the local node# times 3. This was designed historically to stagger the queries to OPER in case many nodes came up at the same time following restoration of electric power.)

So, we have three messages being sent to OPER, but the ARP request is sent only for the first one, with the expectation that reception of an ARP reply will result all sidelined messages being queued and sent to OPER. This means that the ARP request is sent during the initialization of AERS. Sometimes an ARP reply is seen by the Page F diagnostics, and sometimes it is not.

Task initialization after reboot

Look at what happens as the tasks are initialized and brought into execution in an IRM. The tasks are created in numerical order of task# by the ROOT task that operates at a high priority compared to all other tasks that operate at an identical, but lower, priority. Once the ROOT task completes, all tasks start up, each one in turn until it blocks. The Update task (#6), once it has completed its initialization, enters its Forever loop, at the top of which it awaits a task event that is sent by the 15 Hz interrupt routine. Once it receives this event, it performs its usual 15 Hz activities that include interpreting the commands in the Data Access Table that serve to update the data pool with fresh readings. Part of that job involves allowing each local application that has an enabled entry in the local application table (LATBL) to have a chance at the CPU. Two entries are important to this discussion. One is the AAUX entry, and the other is the AERS entry that follows it.

As the AAUX program is initialized, it queues an Acnet ping message to the network, targeting OPER as stated above. Since this is the first message queued for delivery to OPER, an ARP request is sent to obtain the IP address of OPER. While this is going on, additional entries in LATBL are processed. Since the first time involves creating a dynamic memory copy of each local application and invoking it to do its own initialization, the entire procedure can require significant time.

The last task to be initialized is the SNAP task (#12), which supports the IP family of protocols. As part of its initialization, it creates a message queue that will be used to receive references to IP datagrams that are received by the ethernet interrupt routine. This is the important part, because until that message queue has been created, SNAP will not be able to see any received datagram traffic, including ARP replies.

As the Update task finishes its initialization, depending upon the exact synchronization of the accelerator clock 15 Hz timing, there may or may not be a 15 Hz event already waiting. If there is one waiting, Update will not block but will instead perform its first execution through the 15 Hz logic without giving the later tasks a chance to initialize themselves first. If the arrival of the ARP reply—to the ARP request that is sent following the initialization code in AERS—occurs before the SNAP task is initialized, the ethernet interrupt routine deposits a reference to the ARP reply into a nonexistent message queue, so that the ARP reply is effectively ignored.

In the case that the 15 Hz interrupt has *not* been seen by the time the `Update` task blocks, however, the rest of the tasks are initialized, including the `SNAP` task. When the 15 Hz interrupt occurs, everything is fully ready to process IP datagram reception. Anytime after the `Update` task blocks is ok, because all tasks run at the same priority, so that the `Update` task cannot run again until all tasks finish initialization.

Solution

Once the above explanation was realized, the `AERS` local application was modified so that its initialization code no longer sent an Acnet ping message, with the result that failure to download the tables no longer occurred. The Acnet ping was a historic artifact that preceded the support for sidelining messages targeted to a node for which a physical address has yet to be obtained; it is no longer needed.

What is the real solution to the problem? If each task, following its initialization code, just before entering its Forever loop, invoked `NEXTTASK` to give up the CPU, no task would be able to do any work until all tasks are initialized. As long as task initializations do not perform network activities, which is probably valid, we should be ok.

In more detail, consider the logic of how the `USEDELAY` byte is used in the `ETHPQ` entry. Here is a list of the fields represented in its 8 bits:

Bit#	Meaning
7	Used bit that means the entry has been processed.
6–3	Time-out countdown used to count up to overflow into Bit 7.
2–1	n.u.
0	This non-Classic entry was handled by the ethernet transmit routine.

The time-out logic counts 16 cycles at 15 Hz before it overflows and sets the Used bit, providing an approximate one second time-out. Once an entry is queued to the network, it must "get out the door" in one second, or its entry will be declared done. The worst cases would be when many large datagrams that will be fragmented are queued up, so that a significant period of time is required just for the bandwidth. Even so, one second seems like overkill. We could easily cut the time-out period to 8 cycles and thereby free up Bit 3. Since this queue is normally examined in hexadecimal, confining the time-out period counter to the upper nibble might be helpful, which is a reason for the suggestion to cut the time-out period in half.

Two bits not yet used, even if we do not reduce the time-out period, can be used for two diagnostic indicators. Bit 2 can be used to indicate when an entry is queued with the Used bit already set. (A case when this is done is when an alarm message is received from another node, and the alarm message generation option is enabled in the local node. By passing it through the `ETHPQ` with the Used bit set, the message is not sent to the network, but it is seen by the alarm message generation logic in the `QMonitor` task.) Bit 1 can be used to indicate that a time-out occurred. These two additional indicators may make it easier to analyze what happened with messages that were queued to the network. Finally, Bit 0 is now set by the ethernet transmit interrupt code for cases of non-Classic messages only. It would be better to include Classic message types as well. In this way, that bit can show that any message was actually transmitted on the network.